Google Drive Ownership Transfer Application

A comprehensive Node.js application that enables secure transfer of Google Drive file ownership between different Google accounts. This application uses the Google Drive API v3 and implements OAuth 2.0 authentication for secure access to user files.

🌟 Features

- **Single File Transfer**: Transfer ownership of individual files
- **Batch Transfer**: Transfer multiple files in one operation
- **Authentication Management**: Secure token storage and management
- **File Listing**: View files owned by authenticated users
- **Comprehensive Logging**: Detailed logs for audit trails
- **Error Handling**: Robust error handling with detailed feedback
- **Rate Limiting**: Built-in delays to respect Google API quotas
- **Validation**: Input validation for emails and file IDs

🔧 Prerequisites

Before you begin, ensure you have the following installed:

- **Node.js** (v14.0.0 or higher)
- **npm** (Node Package Manager)
- **Google Cloud Console** account with API access

🚀 Installation & Setup

Step 1: Clone and Install

bash

```bash
# Create project directory
mkdir google-drive-transfer
cd google-drive-transfer

# Initialize the project
npm init -y

# Install dependencies
npm install googleapis readline-sync express dotenv
npm install --save-dev nodemon
```

Step 2: Google Cloud Console Configuration

1. **Create a Google Cloud Project**:
   - Go to Google Cloud Console
   - Create a new project or select existing one
   - Note your project ID
2. **Enable Google Drive API**:
   - Navigate to "APIs & Services" > "Library"
   - Search for "Google Drive API"
   - Click "Enable"
3. **Create OAuth 2.0 Credentials**:
   - Go to "APIs & Services" > "Credentials"
   - Click "Create Credentials" > "OAuth 2.0 Client IDs"
   - Choose "Desktop application"
   - Download the JSON file
   - Save it as `credentials.json` in your project root
4. **Configure OAuth Consent Screen**:
   - Go to "APIs & Services" > "OAuth consent screen"
   - Fill in required information
   - Add test users if needed

Step 3: Environment Configuration

Create a `.env` file in your project root:

env

```
NODE_ENV=development
PORT=3000
GOOGLE_CLIENT_ID=your_client_id_from_credentials_json
GOOGLE_CLIENT_SECRET=your_client_secret_from_credentials_json
GOOGLE_REDIRECT_URI=urn:ietf:wg:oauth:2.0:oob
```

Important: Replace the placeholder values with actual values from your `credentials.json` file.

Step 4: Project Structure Setup

Create the following directory structure:

```
google-drive-transfer/
├── src/
│   ├── auth/
│   ├── services/
│   ├── utils/
│   └── app.js
├── config/
├── tokens/
├── logs/
├── .env
├── .gitignore
├── credentials.json
└── package.json
```

📋 Usage Guide

Running the Application

bash

```
# Production mode
npm start

# Development mode (with auto-restart)
npm run dev
```

Step-by-Step Usage

1. **Start the Application**:

```bash
npm start
```

2. **Choose Transfer Type**:
   - Select option 1 for single file transfer
   - Select option 2 for multiple files transfer
3. **Authentication Process**:
   - Enter source account email
   - Enter target account email
   - Follow OAuth flow for each account (browser authorization)
4. **File Selection**:
   - For single files: Enter the Google Drive file ID
   - For multiple files: Enter file IDs one by one
5. **Confirm Transfer**:
   - Review transfer details
   - Confirm to proceed

Finding Google Drive File IDs

Method 1: From URL

- Open the file in Google Drive
- Look at the URL: `https://docs.google.com/document/d/FILE_ID_HERE/edit`
- Copy the FILE_ID_HERE part

Method 2: Using the Application

- Use option 3 in the main menu to list files
- File IDs will be displayed with each file

Authentication Flow

1. **First Time Setup**:
   - Application will open a browser window
   - Log in to Google account
   - Grant necessary permissions
   - Copy authorization code back to application

2. **Subsequent Uses**:
   - Application will use saved tokens
   - Re-authentication only needed if tokens expire

🔐 Security Considerations

Token Management

- Tokens are stored locally in the `tokens/` directory
- Each user's tokens are saved separately
- Tokens include refresh tokens for long-term access

API Permissions

The application requests these Google Drive scopes:

- `https://www.googleapis.com/auth/drive` - Full Drive access
- `https://www.googleapis.com/auth/drive.file` - File-specific access
- `https://www.googleapis.com/auth/drive.metadata` - Metadata access

Rate Limiting

- Built-in delays between API calls
- Respects Google's quota limits
- Configurable delay settings

🔧 Configuration Options

Transfer Settings

```javascript
// In your transfer calls, you can customize:
{
  delayBetweenTransfers: 1500,    // Delay in milliseconds
  continueOnError: true,          // Continue batch on individual failures
  sendNotificationEmail: true,    // Notify new owner via email
  transferOwnership: true         // Actually transfer ownership
}
```

Logging Configuration

- Logs are stored in `logs/` directory
- Daily log rotation
- Different log levels: INFO, SUCCESS, WARN, ERROR, DEBUG

🛠️ Troubleshooting

Common Issues

"Error: invalid_grant"

- Solution: Delete tokens and re-authenticate
- Command: `npm run clean:tokens`

"Error: insufficient_permissions"

- Solution: Ensure proper OAuth scopes are configured
- Check that the user has ownership of the files

"Error: quotaExceeded"

- Solution: Implement longer delays between requests
- Consider running transfers during off-peak hours

"File not found"

- Solution: Verify file ID is correct
- Ensure file hasn't been deleted or moved

Debugging

Enable debug logging:

```bash
NODE_ENV=development npm start
```

View recent logs:

bash

```
# View today's logs
tail -f logs/app-$(date +%Y-%m-%d).log

# View all logs
cat logs/*.log
```

🔍 API Limits and Best Practices

Google Drive API Quotas

- **Queries per day**: 1,000,000,000
- **Queries per 100 seconds per user**: 1,000
- **Queries per 100 seconds**: 10,000

Best Practices

1. **Batch Operations**: Use batch transfers for multiple files
2. **Error Handling**: Always handle API errors gracefully
3. **Rate Limiting**: Implement delays between requests
4. **Token Management**: Store and refresh tokens properly
5. **Logging**: Maintain audit trails for transfers

🧪 Testing

Manual Testing

bash

```
# Test authentication module
npm run test:auth

# Test configuration
npm run test:config

# Test with a single file
npm start
# Choose option 1, follow prompts
```

Validation Testing

The application includes comprehensive validation:

- Email address format validation
- Google Drive file ID format validation
- Configuration completeness validation

📊 Monitoring and Logs

Log Files

- Location: `logs/app-YYYY-MM-DD.log`
- Format: JSON structured logs
- Rotation: Daily automatic rotation

Log Levels

- **INFO**: General application information
- **SUCCESS**: Successful operations
- **WARN**: Warning conditions
- **ERROR**: Error conditions
- **DEBUG**: Debug information (development only)

Sample Log Entry

```json
{
  "timestamp": "2024-06-20 10:30:45",
  "level": "SUCCESS",
  "message": "Transfer operation completed",
  "data": {
    "successful": 5,
    "failed": 0,
    "total": 5
  }
}
```

🔄 Maintenance

Regular Maintenance Tasks

1. **Clean Old Logs**:

```bash
npm run clean:logs
```

2. **Update Dependencies**:

```bash
npm update
```

3. **Remove Expired Tokens**:

```bash
npm run clean:tokens
```

4. **Check API Quotas**:
   - Monitor usage in Google Cloud Console
   - Set up quota alerts if needed

📚 Advanced Usage

Programmatic Usage

```javascript
const GoogleDriveTransferApp = require('./src/app');
const { TransferService } = require('./src/services/transferService');

// Create app instance
const app = new GoogleDriveTransferApp();

// Use individual services
const transferService = new TransferService(sourceAuth, targetAuth);
await transferService.transferFileOwnership(fileId, newOwnerEmail);
```

Custom Integration

The application is designed to be modular. You can import and use individual components:

```javascript
// Authentication only
const GoogleAuth = require('./src/auth/auth');

// Drive operations only
const { DriveService } = require('./src/services/transferService');

// Validation utilities
const ValidationUtils = require('./src/utils/validation');
```

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Add tests if applicable
5. Submit a pull request

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

⚠️ Important Notes

- **Backup First**: Always backup important files before transferring ownership
- **Test Thoroughly**: Test with non-critical files first
- **Monitor Quotas**: Keep track of API usage to avoid hitting limits
- **Security**: Never commit credentials or tokens to version control
- **Compliance**: Ensure compliance with your organization's data policies

📞 Support

For issues and questions:

1. Check the troubleshooting section
2. Review the logs for error details
3. Consult Google Drive API documentation
4. Open an issue in the repository

Happy transferring! 🚀